Reliable Collection of 15Hz Data

Problem for Acnet console

Thu, Apr 20, 2000

The RETDAT data acquisition protocol used by Acnet consoles supports 15Hz data collection in principle, but not in fact. This note describes the problem. Although a new data acquisition paradigm is under development for Java-Acnet, it will be some time before it is ready, so that RETDAT may continue to be used for a long time.

Protocol

The RETDAT protocol allows specifying a reply period in units of 60Hz, with the understanding that only values ≥ 4 are valid. If we use a Frequency Time Descriptor of 4, we can expect to receive 15Hz updates from a front end. The other way to elicit 15Hz replies from a front end is to use an FTD that specifies a clock event that occurs at 15Hz.

Result

Under either scenario, the console *does* get 15Hz replies from the front end, but an application running on the console Vax is unable to receive this data reliably. One reason is that the application polls the data pool for data. An executing application program is scheduled to run every 15Hz cycle, approximately, by using the scheduling resources available from VMS. (The application is scheduled for an execution at successive intervals of 60, 70, and 70 ms, in order to approximate 15Hz execution.) Within each of these executions, the application can poll the data pool for the latest values that it holds for each device/property of interest. When a *set* of devices is being polled via the appropriate CLIB function, a status return is provided to the caller that indicates whether any of the device values of the devices in the set were updated during the time that the entire set was being sampled from the data pool. (A new reply update can occur even during the middle of this sampling.) An application interested in correlated data, *assuming that it knows that the values of all devices in the set come from the same front end in a single reply*, might then choose to repeat the call so as to obtain a set of correlated values.

But there is a problem with the polling scheme, in that two successive updates might arrive from the front end between two scheduled executions of the application, especially if they occur 70 ms apart. There is no timing synchronization at all of the console activity with the operation of the accelerator, so this situation is bound to occur at times.

Callback functions

Is there a method of improving the situation so that 15Hz data can be reliably collected from the 15Hz accelerator? One approach would be to provide callbacks. When a new value arrives from a front end for a given device for deposit into the console data pool, a callback function, having previously been specified by an application during its request initialization, could be invoked to alert the application that fresh data is available for that device. The callback function, then, might queue up data values for processing during its normal "15Hz" execution. This method would make it possible for an application to at least see all of the data that came from the front end.

There is also a sequence number available to the caller to identify which reply from the front end just arrived. This would allow callbacks for separate devices to correlate the various fresh data values that they capture, or at least to learn that they are not correlated. Again, in the case that it is known that all devices of interest originate from the same front end, one might choose to specify a callback function for only one of the devices. When that callback function is invoked, the application could assume that the entire set of device data has been updated and is ready for sampling.

Higher application execution rate

In the absence of support for callback functions, is there a scheme that could work to allow an application to "catch" all 15Hz data? One could consider changing the application scheduling so that it executes at an approximate rate of 30Hz—using delays of 30, 30, 30, 40, 30, and 40 ms. That should allow the application to capture all of the 15Hz data, knowing that it would often receive return status that indicates that the set of data values is the same as that last delivered; perhaps the sequence number is the same as that of the previous set. But this kind of system-wide change would likely confuse a large number of application programs that expect roughly 15Hz execution.

Front end queuing

If the rate of console application execution remains at about 15Hz, perhaps the request could specify to the front end that replies should be delivered at 7.5Hz, or every two cycles, with the front end delivering twice as much data in each reply. (Actually, the D0 detector control system used in Run I included this feature in its controls data acquisition protocol.) In this way, other established applications would not be affected. The console application would need to expect two sets of data in each reply. This means that the data would not be arriving as the usual atomic values of 1/2/4 bytes in length, but rather the reply would consist of a data structure that the application would have to break down to deal with the two sets of data appropriately. Of course, the front end would have to be prepared to make available such multiple sets of data as are needed by the application.

Case in point

It is desirable to collect 15Hz waveforms data for Booster studies that are designed to reduce beam losses in preparation for the heavier use of the Booster that will be required to support MiniBooNE, NuMI, and traditional Tevatron operation. The Booster may operate in bursts of beam cycles, and each must be monitored for analysis. How can we collect 15Hz waveform data using the present RETDAT-based Acnet console system?

Using the idea of buffering in the front end, the waveforms can be copied into data streams, the formalized support for circular buffers that is available in IRMs. The request for data stream data can specify 7.5Hz replies, as described above. The size of the user's buffer must be large enough to hold at least two waveforms, plus a 4-byte header that indicates how many waveforms are actually included in a given reply.

In order to correlate the waveform data with other cycle information, it may be necessary to include more than the bare waveform data in each data stream record. Each IRM maintains a time-of-day to 15Hz accelerator cycle resolution. All IRMs in a single project, synchronized to perform all their 15Hz activities at the same time, are updated with a time-of-day that is obtained every few minutes from a Network Time Protocol server. (This is done because the accelerator's 15Hz, based upon the Booster magnet system operating as a tuned circuit, follows the power line frequency, not the international time standard.) All of one project's IRMs will have the same time-of-day value during the same 15Hz cycle. This time-of-day can be used as a kind of time-stamp to be returned in the data structure. The easiest way to do this is to include the time-of-day in the data stream record. Another useful piece of data is the Booster reset clock event number for that cycle. This scheme can allow the application to associate each waveform with a 15Hz cycle and also to correlate waveforms received from different IRMs.

Synchronized operation

Another scheme that might be considered to provide 15Hz data to a console is to synchronize the console and the front end with the accelerator. But there are many front ends, and each project may use a different synchronization delay to the accelerator. Which front end reference should the console choose for its synchronization? Aside from the difficulty of bringing an interrupt to a console computer, deciding on one synchronization delay that is suitable for all front ends is probably impractical.

Another approach to accelerator synchronization is to key on the arrival of reply data from a front end. This is what is behind the callback scheme described above. Although the console is not synchronized with the accelerator, it can know when reply data has just arrived, which achieves the same goal of access to timely data.

Multiple front ends

Suppose the application requires a set of data that originates from more than one front end. It is not at all clear that there is a solution for this situation in Acnet. Replies from multiple front ends will arrive at unpredictably different times, so that it may be very difficult to determine which reply data goes with which data that comes from another front end.

In the Linac/IRM front ends, support for using a server node to a request is supported. For the case of the Booster HLRF system, all devices from the 22 IRM front ends are declared in the Acnet device DABBEL input to originate from the same front end, which is considered a server node. All Acnet requests for Booster HLRF data pass through a single server node, which uses multicasting to forward the request to the other front ends in that project, then gathers up the reply fragments for subsequent return to the console requester. This means that a single reply message carries data that may have originated from any or all front ends in the project. With all of the data for a request arriving at the same time (in the same datagram) at a console, it is easy to detect correlated data. Although this doesn't solve the problem of access to reliable 15Hz data, it does solve the correlated data problem, which is likely to be important for users that need to collect 15Hz data.

History

Access to reliable 15Hz data has been needed for Linac operation ever since there has been a Fermilab control system. In the earliest days, when one Sigma 2, or a XDS 530, supported one control console for Linac, access to correlated 15Hz data has been provided. As the Linac control system was upgraded in 1982 to a distributed system of seventeen 68K-based nodes, delivery of 15Hz data was maintained effectively by using one node as a server. At some point, the console computes lost the ability to operate with 15Hz accelerator cycles, and the ability to deal with 15Hz data, though still a requirement, required considerable effort at the console application level to maintain. A background-style display of 50 beam currents and beam losses went through much tweaking over the years to display beam data reliably at 15Hz.

Departing from Acnet, the Macintosh-based Parameter Page application program, heavily used to monitor Linac data, has always managed to display reliable 15Hz data. It uses a server node to avoid the problems of sorting out the reply fragments. (It lets the server node do it.) It also uses the Classic protocol to collect data, which is the more efficient native protocol of IRMs.

The front end nodes themselves have always supported a suite of page applications that, acting as clients, gather data from other front ends using the Classic protocol. With all front ends of a project operating in strict synchronization, dealing with 15Hz correlated data is a piece of cake.

Future

The next generation Acnet protocol is expected to deal with 15Hz correlated data, as it is still needed for Linac, and it will be needed for Booster studies, as continuing pressure is placed on the Booster to advance toward 15Hz beam cycles, or at least 7.5Hz. The Tevatron takes beam from Main Injector, which can only get it from Booster. The NuMI project requires beam from Main Injector, which will require more Booster beam cycles. And MiniBooNE takes beam directly from the Booster. Reducing Booster beam losses and continuing to monitor its performance will place demands upon a reliable delivery of 15Hz data from both Linac and Booster.

The new protocol that is to eventually replace RETDAT is called GETS32. It is largely based upon RETDAT, but it is supposed to include a kind of time-stamp that may be a Booster cycle counter in order to achieve 15Hz correlation across front ends. Although this may be the final answer at Fermilab, the new protocol is by no means ready. The RETDAT protocol will have to be used in the interim years until it can be replaced by something better.

Present

This note has described the 15Hz data problem in the Fermilab Acnet control system, especially as seen by an application program at a console. Several possible paths for changes and/or work-arounds were described and evaluated. In order to provide the needed support for accelerator studies, various measures are expected to be used to cope with the problem until the future arrives. It is somewhat ironic that, given that modern computers are several orders of magnitude faster than earlier computers used in this control system, access to reliable 15Hz correlated data remains a challenge.